Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Implementación de una aplicación móvil para una herramienta de gamificación

TITULACIÓN: Grado en Ingeniería de Sistemas Aeroespaciales

AUTOR: Sajda Al Moussaoui

DIRECTORES: Miguel Valero
Roc Meseguer

FECHA: 07 de Febrero del 2020

Título: Implementación de una aplicación móvil para una herramienta de gamificación

Autor: Sajda Al Moussaoui

Directores: Miguel Valero
Roc Meseguer

Fecha: 07 de Febrero del 2020

Resumen

El proyecto desarrollado a continuación se basa en la aportación de una nueva aplicación para cumplimentar los diferentes módulos de una herramienta de gamificación, llamada Classpip, creada hace poco más de dos años por alumnos de la EETAC. Desde entonces, esta herramienta ha tenido muchas mejoras y versiones cada vez más funcionales y manejables para el usuario final y este proyecto es una contribución más para que así sea.

Para llegar a entender el objetivo de este proyecto, primero hay que aclarar el concepto de gamificación que es un concepto clave para la herramienta del Classpip y además conocer la última versión de esta herramienta y sobre qué tecnología se ha construido.

La gamificación es una técnica de aprendizaje que traslada la mecánica de los juegos al ámbito educativo-profesional con el fin de conseguir mejores resultados. Es muy útil para absorber conocimientos, para mejorar habilidades y despertar la motivación de los usuarios. Es un término que ha adquirido una enorme popularidad en los últimos años, sobre todo en entornos educativos y digitales.

En cuanto a la herramienta, actualmente, dispone de 3 módulos diferentes. Una aplicación web y una base de datos desarrollados con la tecnología Angular. Y el tercer módulo es una aplicación móvil que recoge algunas de las funcionalidades importantes de la aplicación web para el perfil del profesor. Este último módulo está desarrollado con la tecnología Ionic.

Este proyecto tiene como objetivo crear un cuarto módulo para cumplimentar el Classpip, desarrollando una nueva aplicación móvil para el perfil del alumno. Esta aplicación pretende ser intuitiva, con funcionalidades útiles para el usuario final, manejable y con la mayor organización posible en el código. También se usará la tecnología Ionic para llevar a cabo esta aplicación, se necesitará la información de la base de datos actual para construir dicha aplicación y también se intentará reutilizar parte del código usado tanto en la aplicación web como en la aplicación móvil del profesor.

Otra de las herramientas necesarias para este proyecto es el Git, donde se encuentran las versiones más recientes de los tres módulos del Classpip que se van a usar como punto de partida para este cuarto módulo. Esta herramienta también será útil para experimentar con diferentes versiones del código y trabajar paralelamente en diferentes secciones del código.

Esta memoria trata de explicar cómo se ha llegado a la conclusión de la necesidad de una aplicación móvil para el perfil del alumno, cómo ha sido todo el proceso de diseño de dicha aplicación y todos los aspectos que se han tenido en cuenta para ello, cómo se ha desarrollado la aplicación y cómo se ha reestructurado el código. Y por último, cómo se ha integrado este nuevo módulo en la herramienta del Classpip.

Title: Implementation of a mobile application for a gamification tool.

Author: Sajda Al Moussaoui

Directors: Miguel Valero
Roc Meseguer

Date: February 7,2020

Overview

The Project developed below is based on the contribution of a new application to complete the different modules of a gamification tool, called Classpip, created just over two years ago by EETAC students. Since then, this tool has had many improvements and versions increasingly functionals and manageables for the end user and this project is a contribution to make it so.

To understand the objective of this project, firstly we have to clarify the concept of gamification which is a keyword for the Classpip tool and also we have to know the latest version of this tool and on which technology has been built.

Gamification is a learning technique that moves the mechanics of games to the educational-professional field in order to achieve better results. It's very useful to absorb knowledge, to improve skills and awaken the motivation of users. It's a term that has gained a lot of popularity in the latest years

The tool, currently, has three different modules. A web application and a database developed with Angular technology. And the third module is a mobile application that collects some of the important functionalities of the web application for the teacher's profile. This last module is developed with Ionic technology.

The aim of this project is to create a fourth module to complete the Classpip, developing a new mobile application for the student's profile. This application aims to be intuitive, with useful features for the end user, manageable and with the best possible organization in the code. Ionic technology will also be used to carry out this application, the information from the current database will be needed and part of the code used in both the web application and the teacher's mobile application will be reused.

Another of the necessary tools for this project is the Git, where the last versions of the three Classpip's modules are to be used as a starting point of this fourth module. This tool will also be useful to experiment with different versions of the code and work in parallel in different sections os the code.

This report tries to explain how the conslusion of the need for a mobile application for the student's profile has been reache, how the entire design process has been done and all the aspects that have been taken into account for it, how the application has been developed and how the code has been restructured. And finally, how this new module has been integrated into the Classpip tool.

ÍNDICE

INTRODUCCIÓN

No cabe duda que la tecnología cada vez es más presente en nuestras vidas. En los últimos años, el uso de ella ha crecido exponencialmente y en todos los ámbitos posibles, se podría decir que la raza humana cada vez es más digital y la enseñanza no es una excepción.

Es un hecho que la tecnología cada vez es más presente en nuestras aulas, los alumnos cada vez son más adictivos a los dispositivos móviles y ya es un estilo de vida para ellos. Todo esto no se puede ignorar y todo el entorno educativo debe adaptarse para poder conectar con este nuevo alumnado. Además de digitalizar las aulas, hay que hacerlo de una forma creativa que despierte el interés de los alumnos, y no hay mejor forma que los juegos.

Con este cambio y esta necesidad de adaptación a la era de la digitalización, nació el concepto de Classpip. Esta herramienta es un entorno gamificado que usa las dinámicas y las mecánicas de los juegos. Este proyecto fue iniciado por distintos alumnos y profesores de la EETAC.

Classpip, actualmente, es compuesto por tres módulos que se complementan. Una base de datos, una aplicación web llamada *dashboard* y una aplicación móvil para el rol del profesor. Este trabajo tiene como objetivo completar el Classpip con una nueva aplicación para el rol del alumno.

El módulo *dashboard* contiene todas las funcionalidades del proyecto, es decir, se pueden crear juegos tanto de puntos como de colecciones, se pueden asignar niveles e insignias a los juegos de puntos y también se pueden crear álbumes de cromos para los juegos de colecciones. De igual manera, se pueden modificar y borrar los juegos. Estos juegos se asignan a los diferentes grupos de clase que tiene el profesor. En cambio, la aplicación móvil del profesor tiene un subconjunto de las funcionalidades que tiene el *dashboard* ya que no todas las funcionalidades son cómodas de ejecutar des del móvil. Por ejemplo, se pueden asignar puntos a un alumno des del móvil, pero no se puede crear una colección de cromos ya que es una funcionalidad más robusta para hacer desde un móvil.

La necesidad de crear una aplicación para el alumno es obvia, en el Classpip hay dos roles fundamentales, el del profesor y el del alumno. Entonces, como ya se dispone de una estructura básica y de una base de datos sólida, se puede desarrollar una aplicación para el alumno. Esta aplicación tiene como objetivo que el propio alumno pueda ver todos sus avances en todos los juegos y que pueda consultar las normas de los juegos en los que participa siempre que quiera.

El contenido de este trabajo se puede dividir en varios capítulos:

En el **capítulo 1**, se introduce el concepto de gamificación en general y luego se focaliza más esta descripción al ámbito educativo. Se justifica por qué a una necesidad de gamificación en las aulas y además de explica un ejemplo real de gamificación en las aulas de un instituto.

Una vez aclarado el concepto teórico, ya se puede profundizar en explicar la herramienta del Classpip. Tanto las tecnologías usadas como los módulos que componen la última versión funcional y esto se desarrolla en el **capítulo 2**.

En el **capítulo 3**, se plantean claramente los objetivos que tiene este proyecto tanto funcionales como de desarrollo sabiendo perfectamente cuán es el punto de partida. Está claro que el objetivo principal es desarrollar una nueva aplicación para el alumno, pero aparte hay que proponerse objetivos más específicos como la estructura del código o las funciones específicas que debe tener esta nueva aplicación.

Teniendo los objetivos claros, el siguiente paso en el **capítulo 4** es diseñar funcionalmente la aplicación. Es decir, diseñar como van a ser cada una de las pantallas, la estructura que van a tener y la información que se va a mostrar en cada una de ellas. Estos diseños se validan con los directores del proyecto y así poder empezar a desarrollar el código de la aplicación.

En el **capítulo 5**, se explicarán todos los estilos gráficos que se han usado en la aplicación para mostrar la información de una manera atractiva y ordenada. También se podrá ver como hay una coherencia de estilo entre todas las pantallas y la fácil navegabilidad que hay entre ellas.

Para entender como se ha desarrollado el código de la aplicación, en el **capítulo 6** se explica cómo se ha estructurado el código, así como qué ficheros se han creado para hacer ese código y como se relacionan entre ellos. También se menciona qué servicios se han usado tanto los propios de Classpip como los proporcionados por el lenguaje de desarrollo.

Una vez implementado todo el proyecto se hacen una serie de pruebas para saber cuánto de robusta es la aplicación a los posibles errores que pueda cometer el usuario. Además, se va a pedir una opinión externa a dos perfiles de persona, una que ya conoce el proyecto y su temática y la otra sería una posible usuaria de la aplicación. Estas pruebas y evaluación se presentan en el **capítulo 7**.

A partir de esta evaluación objetiva y del transcurso de todo el proyecto, en el **capítulo 8** se redactan una serie conclusiones tanto técnicas de la nueva aplicación desarrollada, como personales explicando los retos a los que me he enfrentado así como la manera en como me he enfrentado a ellos y los he resuelto, tal y como haría un ingeniero.

Y por último, en el **capítulo 9**, se hace un anexo como manual para que el usuario pueda descargarse la aplicación y ejecutarla correctamente. También se indica qué tecnologías hacen falta para hacerlo correctamente.

Ahora, ya podemos empezar a explicar el concepto de gamificación.

CAPÍTULO 1. LA GAMIFICACIÓN

El uso de los juegos para motivar el aprendizaje en el alumnado de edades más tempranas ha sido, frecuentemente, estigmatizado en edades más avanzadas al considerarse una pérdida de tiempo. Sin embargo, en los últimos años el fenómeno de la gamificación se ha convertido en una tendencia metodológica con una gran presencia en las aulas. En este capítulo, además de realizar un breve resumen sobre el concepto de la gamificación, también, se describe una experiencia real de dicho concepto en las aulas de un instituto de Barcelona donde se ha utilizado una aplicación implementada *ad hoc* para enseñar, de forma lúdica, contenidos de sistemas digitales en la asignatura de Tecnología Industrial de bachillerato y al final se recogen los resultados de la experiencia y las opiniones de los estudiantes.

1.1. La gamificación como metodología docente

Actualmente, los estudiantes de escuelas, instituto y universidades son nativos digitales. Han crecido utilizando las nuevas tecnologías y presentan nuevas maneras y actitudes ante el proceso de aprendizaje. El profesorado debe hacer frente a nuevos retos para adaptar este proceso a unas nuevas necesidades, además de utilizar diferentes metodologías para conseguir un alumnado participativo, motivado e implicado en su propio aprendizaje.

Es por ello que parece razonable pensar como metodología docente la gamificación. Este término es un anglicismo que puede definirse como el uso de técnicas mecánicas, componentes y dinámicas propias de los juegos y ocio en actividades no recreativas con la finalidad de potenciar la motivación y de reforzar la conducta para solucionar un problema o alcanzar un objetivo. En la **Figura 1.1** se muestra la relación entre estas técnicas mecánicas y dinámicas de los juegos. Las técnicas mecánicas son las maneras de recompensar al usuario en función de los objetivos alcanzados como por ejemplo la acumulación de puntos o la escalación de niveles. Y las técnicas dinámicas hacen referencia a la motivación del propio usuario para jugar y seguir adelante en la consecución de sus objetivos como por ejemplo obtener una recompensa por haber acumulado ciertos puntos en el juego.

Así que, si se aplica una estrategia correcta al introducir un juego dentro de un contexto educativo, aprovechando principios de recompensa, estatus, interacción o competitividad, se pueden fomentar ciertas acciones del estudiante incentivando un comportamiento determinado.

Figura 1.1 Relación entre las mecánicas y dinámicas del juego

1.2. La gamificación en práctica

Son incontables las experiencias de *gamificación* en las aulas, llevadas a cabo en distintos niveles educativos, con el apoyo de las TIC. A continuación, se cita una de ellas seleccionada por su carácter innovador. El siguiente ejemplo pretende presentar y aclarar los conceptos básicos que hay en juego en un entorno de gamificación.

1.2.1. Gamificación en la asignatura Tecnología Industrial del Instituto *Quatre Cantons del Poblenou*

EL primer trimestre del curso 2016-2017, el alumnado que cursa la asignatura de Tecnología Industrial I en primero de bachillerato ha recibido el encargo de organizar y tomar parte en la “Primera carrera de robots seguidores del distrito de *Sant Martí*”, concebida bajo el paradigma del *hardware* y *software* libre y para la cuál necesitarán algunos conceptos básicos de sistemas digitales. Por otro lado, el alumnado que cursa Tecnología Industrial 2 en segundo de bachillerato y pretende examinarse de la asignatura de Tecnología Industrial en las pruebas de acceso a la universidad debe resolver un ejercicio obligatorio sobre sistemas digitales.

Ante la situación, se estimó oportuno llevar a cabo una experiencia de *gamificación* con el objetivo de que los estudiantes aprendan y consoliden conceptos básicos de sistemas digitales y lógica combinacional de una forma lúdica

Figura 1.2 Gamificación en el ámbito educativo

La infografía de la **Figura 1.2** sintetiza el diseño del juego. En primer lugar, el objetivo principal del juego es aprender y consolidar los conceptos relacionados con la lógica combinacional. El juego debe motivar a los alumnos e implicarlos en el proceso de aprendizaje. Para el diseño del juego se ha tenido presentes dos aspectos. 1) La presencia de elementos comunes en las soluciones *gamificadas* como son los puntos de experiencia, los niveles, los *rankings*, las insignias y los retos de dificultad creciente. 2) Se ha tenido en cuenta el tipo de alumnado a quien va dirigido, la experiencia de usuario mediante la utilización de la TIC, la información de su evolución en el proceso de aprendizaje y *feedback* continuo para mejorar las soluciones de los ejercicios.

Aunque, como se ha descrito anteriormente, el objetivo es “aprender de forma lúdica”, no tenemos que olvidar que el alumno “debe aprender”. Una actividad está formada por 4 ejercicios consecutivos. Un ejercicio consiste en la definición de una función lógica en formato de tabla de verdad o expresión booleana. El estudiante debe proponer un circuito combinacional con un conjunto de puertas predeterminadas que define el profesor en el momento de generar el enunciado. Actualmente se dispone de tres niveles de dificultad: inicial, intermedio y experto.

Es importante que el estudiante tenga algún *feedback* en caso de error para poder mejorar el circuito diseñado. El objetivo es que el alumno analice el error y mediante un proceso de prueba y error pueda obtener la solución correcta.

El juego se ha diseñado con las siguientes especificaciones:

- Juegos cortos de 4 ejercicios consecutivos.
- Definición de tres niveles de dificultad.
- Consulta de la clasificación.
- Consulta de la evolución personal.
- Configuración de la privacidad de datos.

Figura 1.3 Menú principal

El juego se ha implementado como una aplicación web que se puede observar en la **Figura 1.3**. El alumnado participante accede mediante un nombre de usuario y una contraseña. Desde el menú principal, el estudiante puede comenzar un nuevo juego, consultar la clasificación (*ranking*), su evolución personal y configurar la privacidad de datos.

Los diferentes niveles de juego siempre están activos. De esta forma, un estudiante puede probar desde el primer acceso los diferentes niveles de dificultad disponibles.

La mecánica del juego es: consiste en ir implementando circuitos lógicos con las puertas lógicas facilitadas con la particularidad de que hay un cronómetro activo. Si se implementa un circuito correcto, el usuario obtiene 100, 125 o 150 puntos (a los que se restan el número de segundos consumidos en implementar el circuito) dependiendo del nivel de dificultad.

Si el circuito implementado es incorrecto, se restan 50, 75 o 100 puntos, (dependiendo de si el ejercicio corresponde al nivel inicial, medio o experto) y se proporciona la tabla de verdad del circuito propuesto por el alumno comparada con la requerida, de manera que se puedan observar las combinaciones incorrectas. Al superar un nivel determinado, la puntuación se almacena en el sistema.

Como hemos descrito, el juego tiene otras opciones de consulta. La opción "*ranking*" permite consultar la clasificación, donde aparecen las 15 máximas puntuaciones obtenidas por los miembros participantes. La opción "evolución personal" muestra el número de participaciones totales, la puntuación acumulada, la media de puntos obtenidos y su nivel que depende del número de participaciones en el juego.

Uno de los grandes agentes motivadores en las soluciones *gamificadas* es el premio, ya sea ofrecido en forma de reconocimiento (como por ejemplo la aparición en las posiciones altas en la tabla de clasificación) o en forma de recompensa. En este sentido, antes de comenzar a jugar, los estudiantes fueron informados de la recompensa que obtendrían: "Quien complete, al menos, 4 juegos en cada uno de los tres niveles, obtendrá medio punto extra en la nota final del trimestre. También se premiará con un punto extra en la nota final a quien obtenga una de las tres puntuaciones medias más altas habiendo jugado, al menos, doce veces".

Después de tres semanas de juego, siete estudiantes habían completado el número mínimo de participaciones necesarias para recibir la recompensa. De estos siete, una alumna prefirió centrarse en conseguir la media más alta y tan solo ha participado en los niveles intermedio y experto, por lo que recibió un punto de recompensa. Dos alumnos recibieron 1,5 puntos: medio punto por completar, al menos, cuatro juegos de cada nivel y un punto por haber registrado la segunda y la tercera posición del ranking. Finalmente, cuatro estudiantes recibieron medio punto de recompensa. El resto no recibió ninguna recompensa al no haber completado el número de ejercicios mínimos requeridos.

Finalmente, se implementó un cuestionario para conocer la opinión de los alumnos del cuál se han extraído muy buenas sensaciones. En cuanto al juego "Aprender lógica combinacional jugando" lo consideran entretenido (73%), fácil de jugar (80%), de aspecto agradable (70%), con un número de niveles adecuado (75%) y que está bien organizado (90%). Al parecer, el paso de los segundos del cronómetro hace que un porcentaje elevado se sienta presionado (75%), aunque no tienen, en general, un sentimiento de frustración cuando fallan (30%). También afirman que han aprendido con el juego (78%), cumpliéndose sus expectativas, y que el *feedback* proporcionado les ha resultado de utilidad (78%).

Figura 1.4 Estado de ánimo de los estudiantes según la encuesta

El análisis del estado de ánimo se puede observar en la **Figura 1.4**. Mientras jugaban, según sus palabras, se han sentido animados, contentos, tranquilos y optimistas. Los sentimientos relacionados con haberse cansado y sentirse hartos no tienen una valoración muy alta. Casi la totalidad del alumnado que ha respondido al cuestionario cree que el juego les ha ayudado a consolidar los contenidos básicos de la lógica combinacional (85%) y han entendido que las tablas se pueden sintetizar en circuitos lógicos (93%). Han entendido cómo aplicar teoremas del álgebra de Boole y leyes de De Morgan para resolver algunos ejercicios (78%) y han entendido que una función lógica, una tabla de verdad y un circuito lógico son tres maneras diferentes de representar la misma información (95%).

Cabe aclarar que el ejemplo que se acaba de explicar es un ejemplo muy concreto y particular, además está pensado para una asignatura en concreto y dirigido hacia un alumnado específico. En cambio, el proyecto Classpip es un proyecto versátil y multiuso que se puede usar en cualquier asignatura y para varios rangos de edades del alumnado. En el siguiente capítulo se explica con más detalle la idea del Classpip.

CAPÍTULO 2. INTRODUCCIÓN AL CLASSPIP

Classpip es una aplicación de *gamificación* orientada al ámbito educativo. Esta aplicación fue creada por el Departamento de Arquitectura de Computadores de la Escola d' Enginyeria de Telecomunicaciones i Aeroespacial de Castelldefels (EETAC) que pertenece a la Universidad Politécnica de Cataluña. Dicha aplicación se empezó a diseñar en el 2016 como trabajo de fin de grado de un alumno y desde entonces varios alumnos han contribuido a esta aplicación, también, con sus trabajos de fin de grado con la ayuda de los profesores del departamento.

Esta herramienta, siempre, ha tenido como objetivo fomentar el aprendizaje en las aulas de forma lúdica ya que hoy en día, la tecnología cada vez es más presente en nuestro día a día. También, remarcar, que esta nueva manera de potenciar la motivación en los alumnos ha tenido muchos éxito en los últimos años y Classpip se centra en las mecánicas y dinámicas de los juegos para que así sea.

Classpip ha evolucionado mucho desde su primera versión en 2016, tal que ahora dispone de tres módulos que interactúan entre ellos. Estos tres módulos son una aplicación web conocida como *Dashboard* con todas las funciones de la herramienta, una arquitectura de servicios como base de datos y una aplicación móvil para el perfil de docente que se presentó en el último proyecto realizado.

2.1. La tecnología del proyecto

Como hemos mencionado antes, Classpip dispone de tres módulos que interactúan entre ellos. Esto es gracias, a la arquitectura desarrollada a la hora de programar la herramienta.

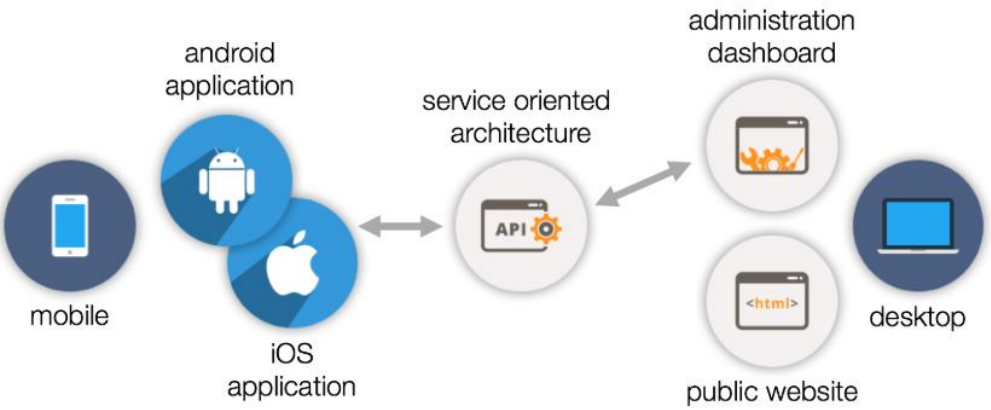


Figura 2.1 Arquitectura del software del Classpip

En la **Figura 2.1** se aprecia perfectamente, que el corazón del proyecto es el módulo de servicios, es decir, la base de datos API REST de donde se nutren tanto la aplicación web como la aplicación móvil. Lo que nos permite la API REST es la interacción con otros módulos, tener una serie de funciones ya definidas para esta interacción sin tener que programarlas manualmente y además poder hacer operaciones desde otros módulos como por ejemplo borrar algún dato solo llamando a la operación DELETE o crear uno llamando la operación POST. En el anterior proyecto, esta herramienta se gestionó con la interfaz *Strongloop LoopBack 3*. Pero en el actual proyecto se ha descargado la última versión que es el *Strongloop LoopBack 4* y es completamente compatible.

Por un lado, está la aplicación web que está diseñada para operaciones y funciones más “complejas”. Este módulo ha sido desarrollado con la tecnología *Angular*, un *framework* que se usa para el desarrollo web y para la ejecución de dicha aplicación se usa *NodeJs*. Y para estilizar esta aplicación y hacerla más amigable hacia el usuario, se ha optado por la interfaz de *Bootstrap*.

Por otro lado, está la aplicación móvil dirigida al rol del profesor que tiene como objetivo realizar funciones más “rápidas”. Para desarrollar el código, también se ha usado *Angular* y para conectar el código con la vista de la aplicación móvil, se ha usado *Ionic*. Pero para poder ejecutar la aplicación tanto en sistema Android como en sistema iOs, se necesita otro *framework* de desarrollo de código libre llamado *Cordova*.

Para poder visualizar, desarrollar o modificar el código de estos módulos se ha hecho a través de la aplicación *Visual Studio Code.* Y al ser un proyecto tan extenso con la participación de varios desarrolladores, el control de versiones se ha hecho a través de la aplicación GitHub. Esta herramienta permite guardar varias versiones del proyecto en la nube y así poder compartirlas y recuperarlas cuando queramos.

2.2. Descripción del Classpip

Classpip se ha creado para que un profesor pueda crear y gestionar juegos tanto individuales como por equipos de los grupos en los cuáles imparte clases. El profesor puede crear varios juegos en un mismo grupo y de varias temáticas para que el alumnado no se acabe cansando de un mismo tipo de juego.

Actualmente, Classpip dispone de **tres** tipos de juegos que un profesor puede usar para motivar el aprendizaje de los alumnos, el juego de puntos, el juego de colecciones y el juego de competición. El objetivo es que cada vez haya más variedad de juegos, por eso paralelamente a este proyecto, otros compañeros están desarrollando el juego de cuestionarios. En la **Figura 2.2** se puede apreciar los tipos de juegos que tiene disponibles el profesor a la hora de crear un juego.

Figura 2.2 La variedad de juegos en el *dashboard*

A la hora de crear un juego, el profesor podrá determinar si el juego será individual o por equipos y crear estos equipos según crea conveniente. Además, estos juegos presentan la opción de añadir galardones a los alumnos por ejemplo el hecho de llegar a un nivel determinado u obtener una insignia.

Además, la aplicación puede tener otras funcionalidades para el profesor como por ejemplo pasar lista.

2.2.1. Juego de puntos

El juego de puntos consiste, básicamente, en asignar puntos a los alumnos individualmente o en quipos según la modalidad del juego. La asignación de puntos se rige por el tipo de puntos que se pueden asignar, es decir, hay diferentes tipos de puntos que el profesor puede asignar a los alumnos según las actitudes o los logros de éstos. Actualmente se disponen de cinco tipos de puntos para este juego.

- Comportamiento
- Puntualidad
- Deberes
- Buena nota en el examen
- Ayudar a un compañero

Además de estos 5 tipos, el profesor puede añadir cualquier otro tipo de puntos según crea conveniente para su grupo.

El juego, también, dispone de niveles que el profesor tiene que crear a la hora de crear el juego tal y como se muestra en **la Figura 2.3**. Estos niveles sirven para otorgar ciertos privilegios a aquellos alumnos que los logren alcanzar. Por lo tanto, es una manera más de motivar al alumno y conseguir que quiera tener más puntos y ponerse en lo más alto de la clasificación.

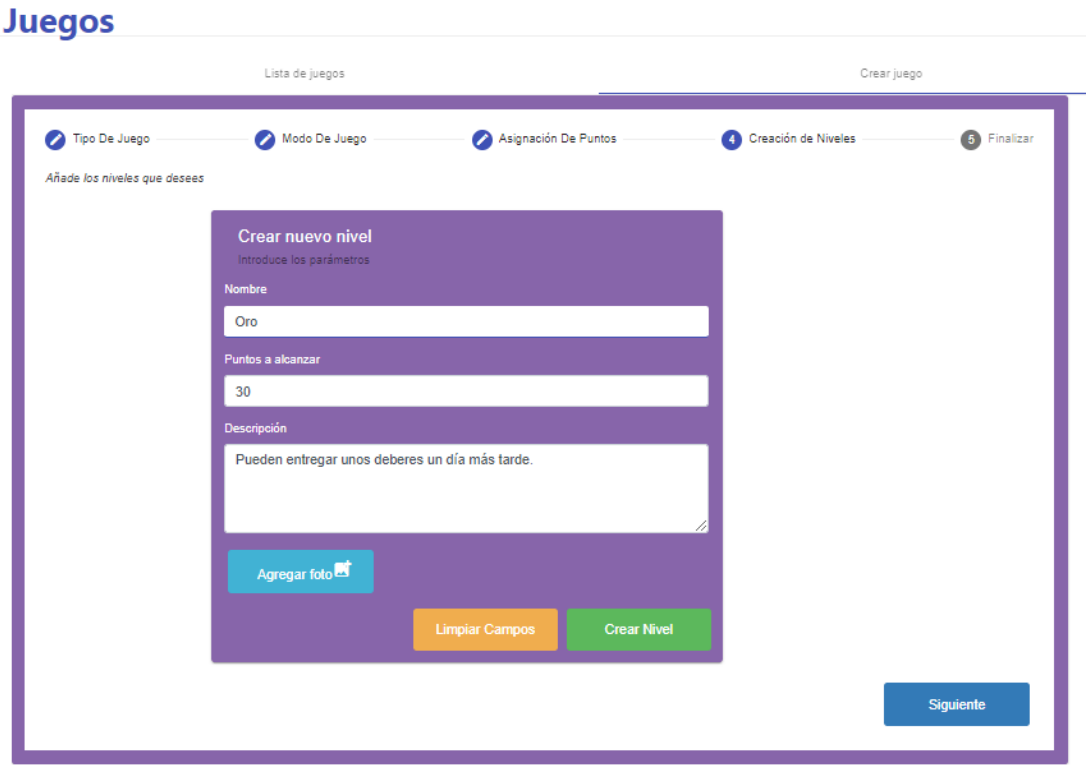


Figura 2.3 Creación de niveles para un juego de puntos

Una vez creado el juego, definiendo si será individual o por equipos y creando los niveles pertinentes a alcanzar por el alumno, el profesor ya podrá asignar los puntos a los alumnos y tener una visión global de cómo va el juego a través de una clasificación donde se muestran el total de puntos por alumno o por equipo y también los niveles que ha logrado alcanzar cada uno. En la **Figura 2.4** se muestra un ejemplo de clasificación. También se puede apreciar que el profesor puede filtrar por el tipo de punto logrado y así ver cómo van sus alumnos en todos los aspectos.

Figura 2.4 Clasificación del juego de puntos

2.2.2. Juego de colección

El juego de colección consiste en asignar cromos a los alumnos individualmente o en quipos según la modalidad del juego. El objetivo principal es tener todos los cromos de la colección del juego. Dicha colección debe ser creada antes de crear el juego. Una vez creada la colección con el número de cromos que el profesor crea conveniente, ésta podrá ser usada para un juego de colección.

Por ejemplo, una colección podría ser de un grupo de físicos, donde cada cromo es un físico diferente. Esto podría ser divertido y a la vez educativo para los alumnos y al final se sepan de memoria un grupo de físico importantes de nuestra historia como podría ser Isaac Newton.

Además, cada cromo puede tener un "nivel" más o menos importante y una probabilidad de asignación más grande o más pequeña según crea conveniente el profesor a la hora de añadir el cromo a la colección (**Figura 2.5**). Por consiguiente, el profesor podría poner como recompensa algún cromo determinado por alcanzar objetivos.

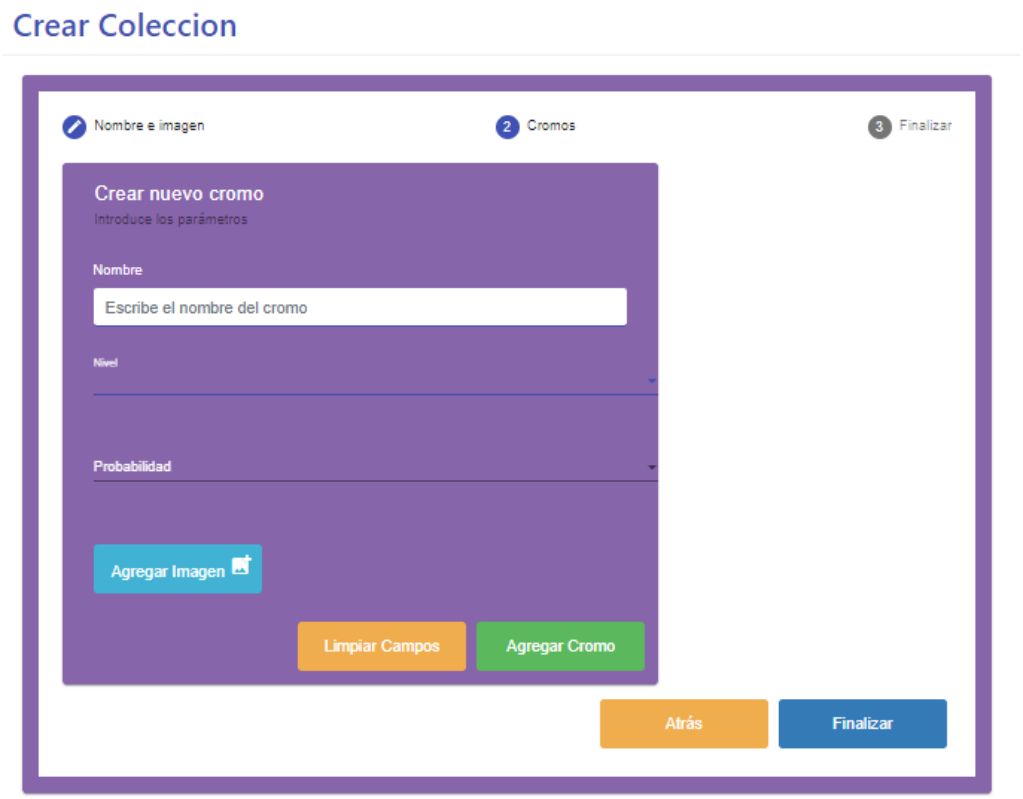


Figura 2.5 Creación de una colección y sus cromos

Una vez el juego de colección está creado y operativo, el profesor puede acceder al álbum de cada alumno o equipo para ver la visión global de cuántos cromos tiene y cuántos cromos le quedan para completar la colección tal y como se aprecia en la **Figura 2.6**. Aquí se muestra un ejemplo de colección de los jugadores de la NBA, pero naturalmente cada profesor podrá crear colecciones más acordes a las temáticas de sus asignaturas, un ejemplo podría ser los filósofos más importantes de la historia de la humanidad.

Figura 2.6 Álbum de un alumno en el juego de colección

2.2.3. Juego de competición

Este tipo de juego aún está en fase de desarrollo para el Classpip, pero su objetivo es hacer un juego cuya clasificación sea igual que la que se usa en los deportes y así tener "partidos" para poder "enfrentar" a los alumnos. Existirán dos tipos de clasificación:

- **Liga**: Tal y como indica su nombre habrá una clasificación tipo la de la liga. Eso permitirá hacer enfrentamientos entre dos alumnos, la idea es que cada semana haya un enfrentamiento y la asignación del premio al vencedor que va a criterio del profesor. Al final del juego, todos los alumnos se habrán enfrentado con todos sus compañeros. El objetivo es quedar lo más alto posible en la clasificación. Cada jornada habrá un enfrentamiento entre dos participantes. El ganador de la jornada se decidirá en base a un criterio decidido por el profesor para esa fecha concreta, podría ser el que mayor nota obtenga en un examen. Este ganador obtendrá puntos para la clasificación en la liga y, además, podría recibir otro tipo de premios para otros juegos en activo (como por ejemplo cromos).

- **Torneo**: En este tipo de juego, los partidos serán eliminatorios hasta que solo quede uno. En cada encuentro, los criterios de la recompensa y el ganador los decide el profesor. A diferencia de la "liga" es que cuando un alumno queda eliminado, ya no puede optar a más recompensas y queda eliminado del juego. Así que se podría decir que este modelo es más competitivo que la "liga".

- **Vuelta ciclista**: Este último tipo también es un juego que tiene jornadas como el de la liga pero la diferencia es que en cada jornada no se juega el uno contra uno sino que es uno contra todos como el Tour de Francia. Los criterios de ordenación se deciden al final por el profesor en base a por ejemplo a una ordenación de las puntuaciones de los exámenes.

En resumen, en el Dashboard se pueden hacer todas las funcionalidades descritas anteriormente. Es decir, se pueden crear juegos de varios tipos y asignarlos a los grupos del profesor. Se pueden crear colecciones de varias temáticas para los juegos de colecciones y se pueden definir niveles e insignias para los juegos de puntos. Una vez creados estos juegos se pueden modificar y obviamente se va a ir asignando cromos y puntos a los alumnos a medida que vayan avanzando en el juego. En cambio, la aplicación desarrollada para el rol del profesor solo puede hacer un subconjunto de las funcionalidades que tiene el Dashboard, es decir, solo aquellas que sean más cómodas realizar desde un dispositivo móvil por ejemplo asignar puntos o cromos a los alumnos en un juego en concreto.

Una vez aclaradas las funcionalidades existentes en el Dashboard y en la aplicación del profesor, se ha creído conveniente que la aplicación del alumno también tenga un subconjunto de funcionalidades que tiene el Dashboard pero desde el punto de vista del alumno. Por ejemplo, que el alumno pueda ver su posición en el ranking de los juegos, pueda ver también los cromos que tiene acumulados en el juego o también ver los de sus compañeros. También sería interesante hacer que el alumno pueda intercambiarse cromos con los compañeros. En el siguiente capítulo se explica más en detalle todas las funcionalidades que pretende tener esta nueva aplicación.

CAPÍTULO 3. OBJETIVOS Y PLAN DE TRABAJO

Una vez conocida la última versión del Classpip, junto con el profesor, decidimos que sería muy útil tener una aplicación para el rol del alumno. Así que el objetivo principal de este proyecto es desarrollar una aplicación móvil para el alumno.

Antes de empezar el desarrollo, hay que tener claro todas las funcionalidades que se pretende que tenga dicha aplicación y son las siguientes:

- Poder listar y ver los detalles de todos los juegos en lo que está inscrito el alumno.
- Poder ver todos los grupos del alumno, así como sus equipos y sus compañeros de clase de cada grupo.
- Visualizar el propio perfil del alumno y poder cambiar pequeños detalles como la contraseña o la imagen de perfil.
- En los juegos de puntos, poder ver la información necesaria sobre los tipos de puntos y el *ranking* actual.
- Poder desglosar los puntos del alumno según el tipo de puntos obtenidos.
- En los juegos de colecciones, poder visualizar todos los cromos obtenidos tanto del propio alumno como el del resto de sus compañeros. O en el caso de ser un juego por equipos, poder ver el álbum del propio equipo y el de los equipos rivales.
- Poder intercambiarse cromos con el resto de compañeros en un mismo juego.
- Obtener una aplicación fácil de usar y fácil de manejar para el usuario final.
- Desarrollar un estilo amigable y agradable.

La idea es que cuando el alumno inicie la sesión en la aplicación, se muestre en la página principal todos los juegos en los que participa el alumno. A parte habrá un menú de navegación des de donde puede navegar el alumno por la aplicación.

A parte de los objetivos funcionales, también hay que tener clara la estructura del código que se va a desarrollar. Un código claro y fácil de entender y de manipular es clave en un proyecto con diferentes módulos y dónde más alumnos puedan aportar sus contribuciones en el futuro.

Partiendo de esta idea, he decidido estructurar el código, de tal manera que haya un *typescript* llamado *services* dedicado solo a hacer consultas a la base de datos, es decir, a la API. Otro *typescript* llamado *calculos*, que se alimenta de *services*, en el cual haré todas las operaciones oportunas para obtener la información final que necesito imprimir en la pantalla de la aplicación. Y por último, habrá el *typescript* de cada pantalla, que lo único que habrá es la declaración de las variables y la llamada de las funciones desarrollados en el *typescript* de *calculos*. En la **Figura 3.1** se muestra este esquema del código.

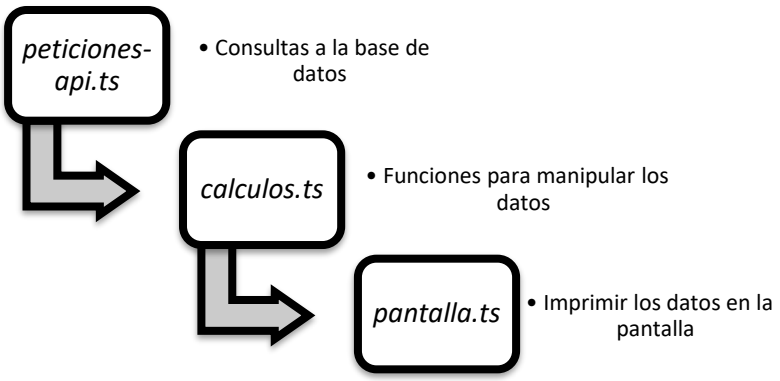


Figura 3.1 Esquema del código

Teniendo en cuenta todos los objetivos mencionados anteriormente, se ha visto conveniente hacer un plan de trabajo para llevar a cabo todo el proyecto. Antes de empezar a pensar en la aplicación en sí, hemos decidido junto con el tutor realizar unas tareas de aprendizaje para familiarizarse con el código y también con la estructura en general del Classpip. Para aprender lo que es puramente código de Angular y de Ionic y también como se trabaja con Github, usaré de guía los vídeos hechos por el tutor y que están colgados en Youtube. Una vez absorbidos los conocimientos de código, el siguiente paso será familiarizarse con la estructura del Classpip, es decir, como está estructurada la base de datos y como se ha organizado el código del Dashboard y de la aplicación del profesor. Para ello el profesor, propone que haga ejercicios de desarrollo directamente sobre el Classpip como por ejemplo pequeñas mejoras que se han detectado que podrían implementarse de una manera sencilla. En global, esta tarea de aprendizaje de código y de Classpip calculamos que podría llevar hasta un mes y medio de dedicación.

El siguiente paso será diseñar las pantallas que va a tener la aplicación del alumno así como todas las funcionalidades que debe tener cada pantalla. Se va a hacer una primera propuesta al tutor y se llegará a una versión final. Esta tarea de diseño y su validación, se calcula que durará entre dos y tres semanas.

Una vez tenga los conocimientos del código y la idea de como va a ser la primera versión de la aplicación del alumno, empezaré a desarrollar el código de esta aplicación. Esta tarea será la más importante y larga del proyecto. Se harán reuniones periódicas con el tutor para ir validando cada avance y corrigiendo posibles errores que vayan surgiendo. Esta fase del proyecto se calcula que podrá extender hasta dos meses de trabajo. En paralelo a esto, empezaré a redactar los primeros capítulos de la memoria.

Y a continuación, una vez tenga la aplicación ya desarrollada del todo y funcionando correctamente, procederé a desarrollar todo el que es el estilo gráfico de la aplicación para hacerla más atractiva de cara al usuario final. Esto me llevará una semana de trabajo.

Y por último, se harán las pruebas pertinentes para testear la aplicación. Primero, yo haré unas pruebas unitarias solo para confirmar que el código no pete y después buscaré a alguien externo para que pueda testear la aplicación y haga una evaluación objetiva. De acuerdo con los resultados de estas pruebas, podré sacar conclusiones sobre la aplicación y encontrar puntos de mejora que se puedan hacer en el futuro. Esta última fase más la tarea de acabar de redactar la memoria se harán en la recta final del proyecto y se calcula que puedan durar de dos a tres semanas de dedicación.

CAPÍTULO 4. DISEÑO FUNCIONAL

Teniendo en cuenta los objetivos mencionados en el apartado anterior, es primordial hacer un buen diseño de la aplicación y así crear y entender una buena estructura de ésta. A continuación, se muestra en qué bloques se organizará la aplicación y el diseño de cada pantalla y su funcionalidad.

4.1. Estructura de la aplicación

La aplicación se dividirá en 4 bloques principales tal y como se muestra en la **Figura 4.1**:

- Mi Perfil
- Mis Grupos
- Mis Juegos Activos
- Mis Juegos Inactivos

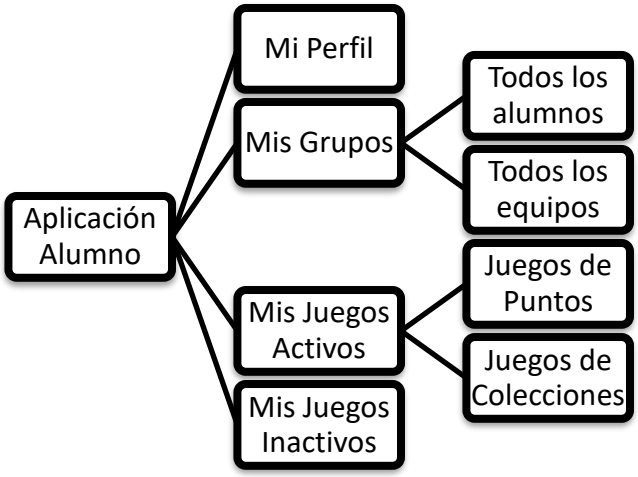


Figura 4.1 Esquema diseño funcional aplicación alumno

4.1.1. Mi Perfil

Permite al alumno acceder a su perfil y poder tener ahí funciones básicas como cambiar su propia foto de perfil o la contraseña.

4.1.2. Mis Grupos

Este bloque es para que el alumno pueda visualizar en qué grupos de clase está asignado. También podrá ver de cada grupo quiénes son sus compañeros y si hay equipos en cuyo grupo, saber qué alumnos componen cada equipo. También se podrá ver quién es el profesor de cada grupo.

4.1.3. Mis Juegos Activos

Este bloque es el más importante de toda la aplicación ya que es el que muestra al alumno todos los juegos activos en los que participa. Principalmente, habrá dos tipos de juegos:

- **Juegos de Puntos**: En cada juego de puntos en el cual participe el alumno:
    - Se mostrará un *ranking* con los puntos de cada alumno o de cada equipo dependiendo si es un juego **individual** o por **equipos**.
    - El alumno podrá acceder a visualizar el detalle de sus puntos o los puntos de su equipo. Es decir, podrá saber en qué cualidades ha obtenido más puntos y en cuales menos puntos.
- **Juegos de Colecciones**: En los juegos de colecciones del alumno:
    - Se podrá visualizar un listado de sus compañeros participantes en el juego si es un juego **individual** o de **equipos** si es un juego por equipos.
    - El alumno podrá acceder a ver sus propios cromos y los de sus compañeros, así como cuantos ejemplares tienen de cada cromo. Esto también se aplica a los equipos si es un juego por equipos.
    - Se podrá apreciar los cromos que aun no tiene el alumno ya que estarán pintados de un color más gris para distinguirlos. Esto le dará perspectiva al alumno para que pueda apreciar cuántos cromos le faltan para completar la colección.
    - Se habilitará una función para que los alumnos puedan intercambiarse cromos entre ellos. Por ejemplo, si un alumno ve que tiene dos ejemplares de un mismo cromo, éste puede entrar en la aplicación y concederle este cromo a otro compañero.

4.1.4. Mis Juegos Inactivos

En este último bloque, el alumno podrá ver un listado de sus juegos que no están activados por el profesor. Aquí no se podrá ver el detalle de ningún juego hasta que el profesor lo haya activado y pase a la pantalla de Juegos Activos.

4.2. Diseño de las pantallas

Para el diseño de las pantallas, se ha usado la plataforma *MockingBot* que se usa para el diseño de aplicaciones móviles. Esta plataforma permite el correcto diseño de forma simple y estructurada de cada pantalla.

Primero se han diseñado las pantallas para los 4 grandes bloques mencionados en el aparatado anterior. Y después, se ha ido profundizando en cada bloque para diseñar todas las pantallas necesarias acorde a las necesidades de cada bloque.

4.2.1. Pantalla de “Mi Perfil”

Esta pantalla es el área privada de cada alumno, es decir, aquí el alumno podrá visualizar sus credenciales de inicio de sesión y también debería poder cambiarlas. Así como visualizar su foto de perfil y también cambiarla. El aspecto de esta pantalla se muestra en la **Figura 4.2**.

Figura 4.2 Pantalla de “Mi Perfil”

4.2.2. Pantalla de “Mis Grupos”

La pantalla del área de “Mis Grupos” se dividirá, principalmente, en dos. Una parte para mostrar los alumnos de cada grupo de clase del alumno y otra parte para los equipos formados en cada uno de estos grupos. El diseño se puede ver en la **Figura 4.3**.

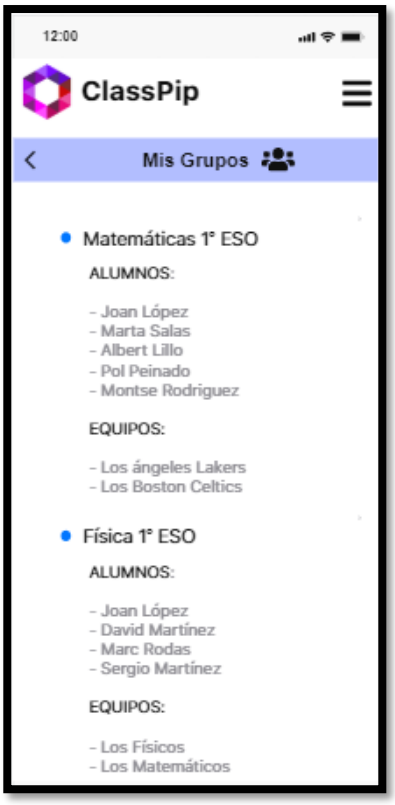


Figura 4.3 Pantalla de “Mis Grupos”

4.2.3. Pantallas de “Mis Juegos Activos”

Al entrar en “Mis Juegos Activos”, se mostrará un listado de todos los juegos que tiene activos el alumno tal y como se muestra en la **Figura 4.4**.

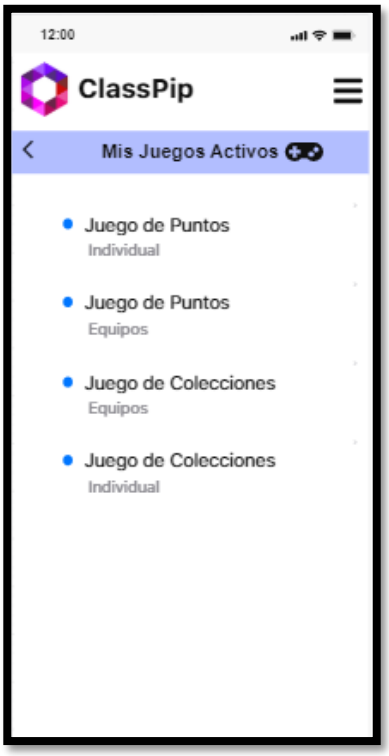


Figura 4.4 Pantalla de “Mis Juegos Activos”

- **Juego de Puntos**: Al hacer *click* sobre algún juego de puntos, se mostrará el detalle de dicho juego cuyo diseño se muestra en la **Figura 4.5**. Si es un juego Individual se mostrará el *ranking* por alumnos y si es de equipos, se mostrará por equipos.

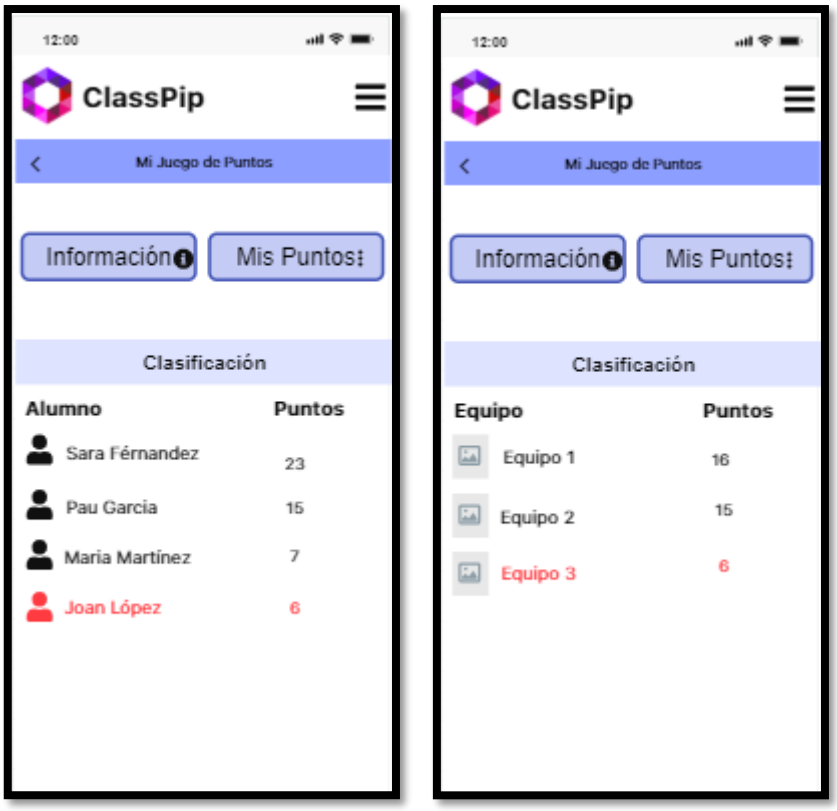


Figura 4.5 Pantallas de “Mi Juego de Puntos”

En la **Figura 4.5** se aprecia que hay dos botones. El que se denomina *Información* es para que el alumno cuando clique sobre él, pueda ver las bases del juego. Y el de *Mis Puntos*, es para que el alumno pueda ver su detalle de los puntos tal y como se ve en la **Figura 4.6**.

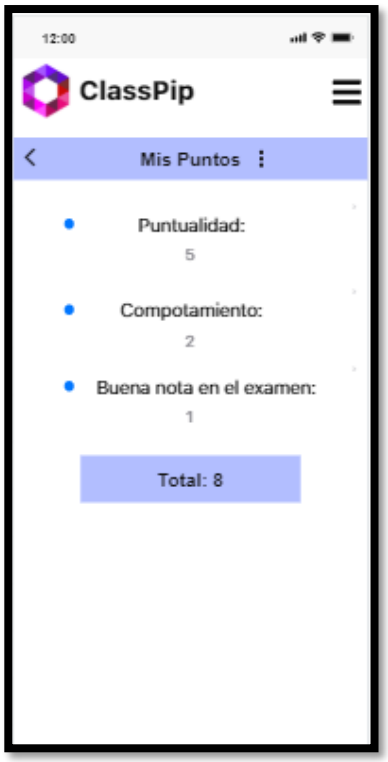


Figura 4.6 Pantalla de detalle de “Mis Puntos”

- **Juego de Colecciones**: Des de la pantalla de “Mis Juegos Activos”, al clicar sobre un juego de colecciones se muestra un listado de los alumnos o equipos que participan en dicho juego. Se puede ver el diseño de dicha pantalla en la **Figura 4.7**.

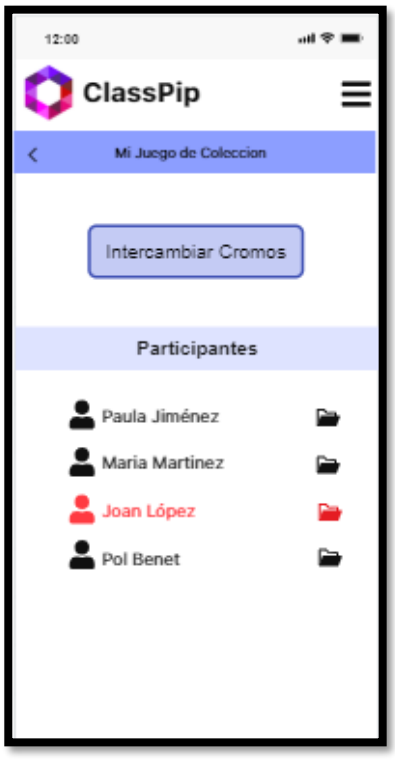


Figura 4.7 Pantalla de “Mi Juego de Colección”

Sobre la pantalla de “Mi Juego de Colección”, se puede navegar hacia dos pantallas más. La primera es cuando se clica sobre algún álbum de los participantes del juego donde se puede ver los cromos que tiene el alumno o el quipo y también los que no tiene. En la **Figura 4.8** se visualiza el prototipo de esta pantalla. Y la segunda pantalla es la que se muestra en cuanto el alumno clique sobre el botón de *Intercambiar Cromos.* En esta pantalla, el alumno podrá elegir uno de los cromos que tiene en su álbum y asignárselo a uno de sus compañeros en el juego. El diseño de esta pantalla se puede ver en la **Figura 4.9**.

Figura 4.8 Pantalla del “Álbum”

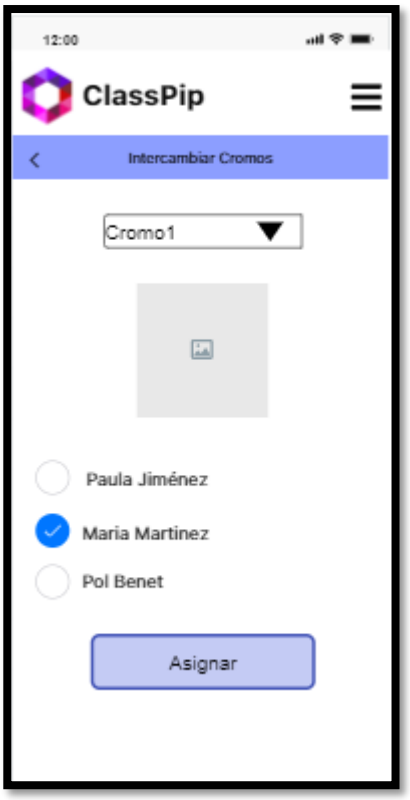


Figura 4.9 Pantalla de “Intercambiar Cromos”

4.2.4. Pantalla de “Mis Juego Inactivos”

Por último, mencionar que la pantalla de “Mis Juegos Inactivos” será muy similar a la de “Mis Juegos Activos”, a diferencia que en estos juegos no se podrá navegar para ver el detalle.

CAPÍTULO 5. ESTILO GRÁFICO

El estilo gráfico es la capa que hay entre el usuario y el corazón funcional de la app, el lugar donde nacen las interacciones. En mayor medida está compuesta por botones, gráficos, íconos y fondos.

Una de las ventajas que tiene definir el estilo de las pantallas con Ionic es que es compatible tanto con Android como con iOS. Pero por otro lado, este diseño al realizarlo por código no es tan intuitivo y además Ionic limita las importaciones de componentes externos por compatibilidad así que a la hora de escribir el código de los estilos, no ha sido tan fácil aplicar el que se ha diseñado con el *MockingBot* en el capítulo 4.

El código del diseño de cada uno de los módulos, se desarrolla en el módulo *scss* de cada pantalla. Y luego, des del módulo *html* se llama a dichos estilos. A continuación, se explica cómo se ha llevado a cabo el diseño de cada uno de los componentes de la aplicación desarrollada. En todo momento, se ha intentado mantener la coherencia entre todos los módulos de la aplicación.

5.1. Cabecera

El diseño de la cabecera es común para todas las pantallas de la aplicación. Así que se ha hecho un diseño único y se ha llamado desde todos los módulos *html* de cada pantalla. En la **Figura 5.1** se muestra un ejemplo de esta cabecera en la pantalla de inicio de la aplicación.

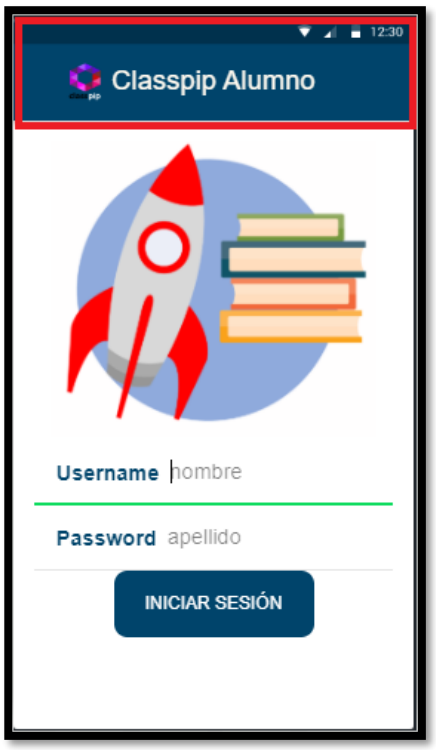


Figura 5.1 Ejemplo de cabecera en la pantalla de inicio

La cabecera está compuesta por un título y una imagen que es el logo del Classpip. La imagen del logo se llama des del *html* y se define su posición y su tamaño en el *scss* con la función *ClasspipCabeceraimg* que también se llama des del *html* con la variable class. Se hace exactamente lo mismo para el título. En la **Figura 5.2** se ve como se llama a las funciones del módulo *scss* con la variable *class* y en la **Figura 5.3** se puede ver como se ha escrito el código de cada función. En esta última figura también se puede ver como se ha dado color a toda la cabecera con la variable *ion-toolbar*.

```
<ion-header>
  <ion-toolbar>
    <ion-title class="titulo">
      Classpip Alumno
    </ion-title>
    <img class = "ClasspipCabeceraimg" src="../../assets/icon/classpip1.png">
  </ion-toolbar>
</ion-header>
```

Figura 5.2 Módulo *html* de la pantalla de inicio

```
ion-toolbar {
    --ion-background-color: rgb(0, 68, 107);
  }

.ClasspipCabeceraimg{
    position:relative;
    width:30px;
    height:30px;
    margin-left: 13%;
    margin-right: -15%;
    margin-top: -10%;
    margin-bottom: 3%;
}
.titulo{
  text-align: center;
  color: floralwhite;
  margin-top: 5%;
}
```

Figura 5.3 Módulo *scss* de la pantalla de inici

5.2. Menú lateral

Se ha creado un menú lateral en la aplicación para facilitar la navegación entre los diferentes bloques descritos en el capítulo anterior. Es decir, las pantallas principales de los juegos activos, los juegos inactivos, los grupos y el perfil del alumno. Además en este menú se ha añadido la función de cierre de sesión. El menú se ha creado des del *html* de la aplicación para que se pueda llamar desde todas las pantallas que queramos. Como se ve en la **Figura 5.4**, tiene acceso a los 4 bloques principales y al cierre de sesión. En el *html* se usa el componente *ion-menu* y para crear cada acceso se usa el *ion-item* de Ionic (**Figura 5.5**).

Figura 5.4 Menú lateral de la aplicación

```
<ion-menu side="start" menuId="first" contentId="content1" color="--ion-color-tertiary-tint">
  <ion-header>
    <ion-toolbar>
      <ion-title style="color: floralwhite;margin-top: 7%; margin-right: 10%;text-align: center;">Menú</ion-title>
      <img class = "ClasspipCabeceraimg" src="../assets/icon/classpip1.png">
    </ion-toolbar>
  </ion-header>
  <ion-content>
    <!--div *ngFor="let Alumno of MiAlumno"-->
      <img src="../assets/icon/UsuarioAlumno.jpg" class="img">
        <ion-label style="text-align: center; font-size: 120%; color:rgb(255, 255, 255)"></ion-label>

      <!--Se define los botones que se visualizarán en el desplegable lateral del usuario-->
      <ion-menu-toggle auto-hide="true">
        <ion-item menuClose ion-item (click)="GoMiPerfil();"><b>Mi Perfil</b>
          <ion-icon item-left class="icon" name="person"></ion-icon>
          <ion-icon item-right class="flecha" style="margin-left: 47%;" name="arrow-forward"  ></ion-icon>
        </ion-item>
```

Figura 5.5 Código del *html* del menú lateral

En el *html* se llama a las funciones desarrolladas en el *scss* para estilizar a los iconos y a las flechas del menú. Como se puede ver en la **Figura 5.5**, estas funciones son *icon* y *flecha.* Sú código se muestra en la **Figura 5.6**.

```
.icon{
  color: rgb(0, 68, 107);
  width:25px;
  height:25px;
}

.flecha{
  color: darken($color: #000000, $amount: 0);;
}
```

Figura 5.6 Funciones del *scss*

5.3. Listas

En muchas pantallas de la aplicación se ha hecho uso de las listas por ejemplo para mostrar todos los juegos en los que participa el alumno (**Figura 5.7**) o para mostrar los niveles de cada juego. Para definir una lista, des del *html* de la pantalla se llama al componente *ion-list*. Y con la variable *class* se llama al scss para definir el estilo (**Figura 5.8**). En el caso del estilo, se ha desarrollado la función *container* en el *scss* (**Figura 5.9**) y así se aplica para cada valor que tenga la lista que en este caso es cada juego activo que tiene el alumno.

Figura 5.7 Pantalla de la lista de juegos activos del alumno

```
<!--Se define el contenido de la tabla de Juegos Activos-->
<ion-list *ngIf="JuegosActivos[0];else Aviso_no_juegos_activos">
    <!--Si no tiene valor JuegosActivos,
    entonces aparecerá la alerta de Aviso_no_Juegos_Activos-->
        <ion-list class="container" *ngFor="let juego of JuegosActivos" (click) = JuegoSeleccionado(juego) >
          <h2 style="text-align: center;">{{juego.Tipo}}</h2>
          <p style="text-align: center;" id="p"> {{juego.Modo}} </p>
        </ion-list>
  <ng-template #Aviso_no_juegos_activos >
        <label class = "AvisoNoJuegosActivo" >
          No hay juegos activos</label>
  </ng-template>

</ion-list>
```

Figura 5.8 *Ion-list* del módulo *html* de la pantalla de juegos activos

```scss
.container {
  z-index: -1;
  position: relative;
  font-family: Righteous;
  width: 100%;
  height: 100px;
  max-width: 100%;
  max-height: 100px;
  background: rgb(191, 234, 240);
  border-radius: 10px;
  border-top-width: 30px;
  margin-top: 1%;
  box-shadow: 2px 1px 2px 2px rgba(0, 0, 0, 0.25);
}
```

Figura 5.9 Función del módulo *scss* de la pantalla de juegos activos

5.4. Tablas

También se ha hecho mucho uso de las tablas en la aplicación, principalmente, para mostrar los rankings de los juegos. En Ionic no existe el concepto tabla como tal, sino que para mostrar una tabla se hace uso de filas y columnas. Es decir, se llama a una fila y luego dicha fila se divide en tantas columnas como necesite el usuario. Esto hace que el diseño se ha aún más complicado.

Para crear la tabla de la **Figura 5.10**, se ha creado una fila única para la cabecera de la tabla y luego por separado tantas filas como alumnos que participan en el juego (**Figura 5.11**). Evidentemente todas las filas, tienen tres columnas. Esto se ha hecho así para poder hacer una función de diseño en el scss para la cabecera llamada *cabeceraTabla* y otra diferente para el resto de filas llamada *casillas* (**Figura 5.12**).

Figura 5.10 Tabla del ranking de un juego de puntos

```
<ion-grid>
  <!--Se define la cabecera de la tabla del ranking del juego de Puntos-->
  <ion-row class="cabeceraTabla">
    <ion-col col-1 style="margin-left: -7%">
      <p ><b>Nombre</b></p>
    </ion-col>
    <ion-col col-2 style="margin-left: -5%">
      <p ><b>Apellido</b></p>
    </ion-col>
    <ion-col col-3 style="margin-left:-5%">
      <p><b>Puntos</b></p>
    </ion-col>
  </ion-row>

  <!--Se define el contenido de la tabla , es decir los nombbres y los puntos-->
  <ion-row class="casillas" *ngFor="let item of MisAlumnosAMostrar">
    <ion-col col-1>
      <p ><b>{{item.Nombre}}</b></p>
    </ion-col>
    <ion-col col-2>
      <p ><b>{{item.PrimerApellido}}</b></p>
    </ion-col>
    <ion-col col-3 style="margin-left:5%">
      <p><b>{{item.PuntosTotalesAlumno}}</b></p>
    </ion-col>
  </ion-row>
</ion-grid>
```

Figura 5.11 Filas y columnas del *html* del ranking del juego de puntos

```
.cabeceraTabla{
  background-color: rgb(0, 68, 107) ;
  margin-top:8%;
  margin-bottom: -2%;
  height:40%;
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
  padding-inline-start: 7%;

  p {
    font-size: 13px;
    line-height: 1.5;
    color: rgb(255, 250, 250);
  }
}
```

```
.casillas{
  background-color:transparent;
  p {
    font-size: 14px;
    color: rgb(0, 68, 107);

    b {
      color: rgb(0, 68, 107) ;
    }
  }

}
```

Figura 5.12 Funciones del *scss* del ranking del juego de puntos

5.5. Botones

Básicamente se ha usado dos tipos de botones en esta aplicación. El primero es un botón blanco que tiene sombras alrededor para cuando haya dos botones uno al lado del otro (**Figura 5.13**). Para posicionar dos botones uno al lado del otro, hay que declarar una fila con dos columnas y cada columna será un botón que llamará a una función del *scss* que es la que definirá el estilo de este botón. Como se puede ver en la **Figura 5.14**, las funciones del scss se denominan *BotonIzquierda* y *BotonDerecha* que se puede ver su código en la **Figura 5.15**. A parte de la declaración del botón con un *ion-item,* se puede apreciar en la **Figura 5.14** que hay un *ion-icon* declarado dentro del *ion-item*. Esto sirve para poner un ícono dentro del botón.

Figura 5.13 Ejemplo de dos botones, uno al lado del otro

```
<ion-row style="text-align: center;">
  <ion-col col-6 >
      <ion-item class="BotonIzquierda" (click) = VerInformacion();> Información
          <ion-icon style="margin-right: 2%;" name="information" item-left ></ion-icon>
      </ion-item>
  </ion-col>

  <ion-col col-6>
      <ion-item class="BotonDerecha" (click) = VerRanking();>Ver Ranking
          <ion-icon style=" margin-right: 2%;" name="more" item-left ></ion-icon>
    </ion-item>
  </ion-col>

</ion-row>
```

Figura 5.14 Ejemplo de *html* de dos botones juntos

```
.BotonIzquierda{
  color: rgb(231, 159, 159);
  font-size: 75%;
  font-style: bold;
  height: 100%;
  max-height: 100px;
  margin-top:0%;
  border-top-left-radius: 10px;
  border-top-right-radius: 10px;
  border-bottom-left-radius: 10px;
  border-bottom-right-radius: 10px;
  box-shadow: 0px 3px 6px 0px rgba(223, 44, 44, 0.966);
}
```

```
.BotonDerecha{
  background-color: rgb(145, 208, 245) ;
  color: rgb(0, 68, 107);
  font-size: 75%;
  font-style: bold;
  height: 100%;
  max-height: 100px;
  margin-top:0%;
  border-top-left-radius: 10px;
  border-top-right-radius: 10px;
  border-bottom-left-radius: 10px;
  border-bottom-right-radius: 10px;
  box-shadow: 0px 3px 6px 0px rgb(0, 68, 107);
}
```

Figura 5.15 Funciones del scss para estilizar los botones

El segundo tipo de botón es para cuando haya un solo botón centrado y estará pintado para destacar. Un ejemplo es el botón que se usará para intercambiar los cromos en los juegos de colección tal y como se ve en la **Figura 5.16**. En este caso no hace falta declarar ninguna fila ni columna en el *html* ya que es un solo botón. Se declara con un *ion-item* (**Figura 5.17**) que llama a la función *scss* que en este caso se llama *IntercambiarCromo* (**Figura 5.18**).

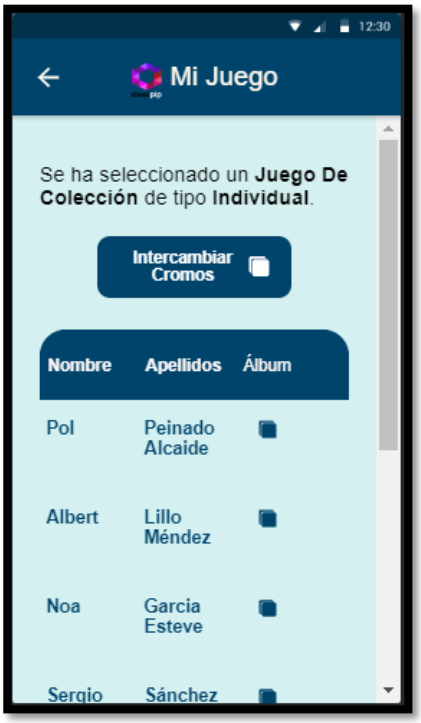


Figura 5.16 Ejemplo de botón centrado

```
<ion-item class="IntercambiarCromo" (click) = IrIntercambiarCromos(); style="text-align: center;">
  <b>Intercambiar Cromos</b>
  <ion-icon style="color:  rgb(255, 255, 255);" name="albums" item-left ></ion-icon>
</ion-item>
```

Figura 5.17 Ejemplo de declaración de un botón centrado en el *html*

```
.IntercambiarCromo{
  --ion-background-color: rgb(0, 68, 107) ;
  color:  rgb(255, 255, 255);
  text-align: center;
  font-size: 80%;
  font-style: bold;
  height: 150%;
  max-height: 100px;
  width:100%;
  max-width: 150px;
  margin-right: -15%;;
  margin-left: 20%;
  border-top-left-radius: 10px;
  border-top-right-radius: 10px;
  border-bottom-left-radius: 10px;
  border-bottom-right-radius: 10px;
}
```

Figura 5.18 Ejemplo de función scss de un botón centrado

5.6. Cromos

En la visualización de un álbum de cromos hay que distinguir entre dos tipos de cromos, los cromos que tiene el alumno y los que no tiene. Se hace esta distinción para que los cromos que tenga el alumno se vean con colores más vivos y en cambio los cromos que no tenga se vean más transparentes o con una capa gris encima. Primero, para posicionar los cromos en la pantalla se ha aprovechado el componente *ion-card* de Ionic en el *html* que permite visualizar una imagen con un título y una descripción abajo. En este caso, el título será el nombre del cromo y en la descripción se mostrará cuantas repeticiones tiene el alumno de cada cromo tal y como se muestra en la **Figura 5.19**. Como el componente *ion-card* ya viene predefinido, solo se han hecho unos retoques en el título y la descripción en el mismo *html* con la variable *style* (**Figura 5.20**) que básicamente es cambiarles el color. Pero para los cromos que no tiene el alumno sí que se ha creado una función en el *scss* llamada *MeFalta* (**Figura 5.21**) para poder visualizarlos más grises que el resto de cromos.

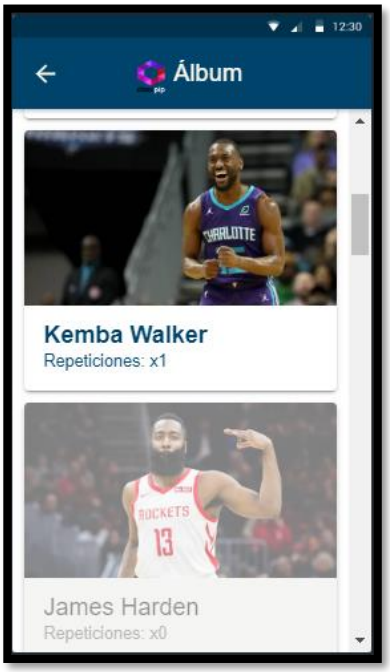


Figura 5.19 Ejemplo de álbum de cromos

```
<ion-card *ngFor="let elem of listaCromosSinRepetidos;let i = index">
  <img [src]="MisImagenesCromo[i]" >
  <ion-card-content>
    <ion-card-title style="color: rgb(0, 68, 107);">
      <b>{{elem.cromo.Nombre}}</b>
    </ion-card-title>
    <p style="color: rgb(0, 68, 107);">
      Repeticiones: x{{elem.rep}}
    </p>
  </ion-card-content>
</ion-card>
<ion-card class="MeFalta" *ngFor="let elem of CromosQueNoTengo;let i = index">
  <img [src]="ImagenesCromosQueNoTengo[i]" >
  <ion-card-content>
    <ion-card-title>
      {{elem.cromo.Nombre}}
    </ion-card-title>
    <p>
      Repeticiones: x{{elem.rep}}
    </p>
  </ion-card-content>
</ion-card>
```

Figura 5.20 Ejemplo de *html* para el álbum de cromos

```scss
.MeFalta {
    opacity: 0.5 !important;
    background-color: rgb(235, 235, 228) !important;
    box-shadow: 0px 3px 6px 0px  rgb(126, 138, 141) !important;
}
```

Figura 5.21 Función *scss* para que los cromos se vean más grises

CAPÍTULO 6. IMPLEMENTACIÓN

Para el desarrollo de la aplicación del alumno se hace uso de gran parte de las clases y servicios del Dashboard. Pero también se han creado funciones nuevas para las funcionalidades específicas de esta aplicación.

Lo que más se ha intentado respetar a la hora de desarrollar la aplicación es el orden descrito en el capítulo 3. Es decir, todo lo que son las consultas y las modificaciones sobre la API se hacen des del servicio de Peticiones a la API. Después, todo lo que sea hacer modificaciones y cálculos sobre esa información obtenida de la API se hace en otro servicio llamado Cálculos. Y luego en el *typescript* de cada pantalla o componente, solo se trata de llamar a estas funciones desarrolladas en el servicio de cálculos. Además de estos dos servicios, se va a hacer mucho uso de otro servicio llamado Sesión que es para guardar constantes que luego podamos necesitar en otro momento, ya solo tendremos que llamar a Sesión para obtenerlas en vez que hacer otra vez el recorrido de Peticiones a la API y luego Cálculos.

Además de los servicios creados por el desarrollador, en esta aplicación se hace uso de servicios proporcionados por Ionic como el servicio de *Router* para navegar de una pantalla a otra o los servicios de *LoadingController* y *AlertController* para crear ventanas emergentes de avisos.

6.1. Peticiones a la API

En este servicio se incluyen todas las solicitudes a la base de datos mediante peticiones *http*, como métodos de obtención, modificación o eliminación de datos. Primero se declaran las constantes que van a llevar las URL's que necesitaremos (**Figura 6.1**) y luego llamaremos a esas constantes desde una función para obtener la información que necesitamos.

```
export class PeticionesAPIService {


  private APIUrlProfesores = 'http://localhost:3000/api/Profesores';
  private APIUrlAlumnos = 'http://localhost:3000/api/Alumnos';
  private APIUrlGrupos = 'http://localhost:3000/api/Grupos';
  private APIUrlMatriculas = 'http://localhost:3000/api/Matriculas';
  private APIRUrlJuegoDePuntos = 'http://localhost:3000/api/JuegosDePuntos';
  private APIUrlAlumnoJuegoDePuntos = 'http://localhost:3000/api/AlumnoJuegosDePuntos';
  private APIUrlLogoEquipo = 'http://localhost:3000/api/imagenes/LogosEquipos/download/';
  private APIUrlEquipos = 'http://localhost:3000/api/Equipos';
  private APIUrlLogosEquipos = 'http://localhost:3000/api/imagenes/LogosEquipos';
  private APIUrlPuntosJuego = 'http://localhost:3000/api/AsignacionPuntosJuego';
  private APIUrlImagenNivel = 'http://localhost:3000/api/imagenes/imagenNivel';
  private APIURLImagenInsignia = 'http://localhost:3000/api/imagenes/ImagenInsignia';

  private APIUrlAlumnoJuegoPuntos = 'http://localhost:3000/api/AlumnoJuegosDePuntos';
  private APIUrlEquipoJuegoPuntos = 'http://localhost:3000/api/EquiposJuegosDePuntos';
```

Figura 6.1 URL's que se usan en el servicio de Peticiones a la API

Un ejemplo muy claro es la comprobación que se hace cuando un alumno intenta acceder a la aplicación poniendo como usuario el nombre y como contraseña el apellido. En ese caso des del componente *typescript* de la pantalla de inicio se llama a la función *DameAlumno* (**Figura 6.2**) que está en el servicio de Peticiones a la API (**Figura 6.3**).

```
this.peticionesAPI.DameAlumno(this.nombre, this.apellido).subscribe(
  (res) => {
    if (res[0] !== undefined) {
      this.alumno = res[0];
```

Figura 6.2 Llamada de una función del servicio de Peticiones a la API

```
public DameAlumno(nombre: string, apellido: string): Observable<Alumno> {
  console.log('Entro a mostrar a ' + nombre + ' ' + apellido);
  return this.http.get<Alumno>(this.APIUrlAlumnos + '?filter[where][Nombre]=' + nombre
  + '&filter[where][PrimerApellido]=' + apellido);
}
```

Figura 6.3 Código de la función del servicio Peticiones a la API

6.2. Cálculos

En este servicio se desarrollan las funciones necesarias para hacer modificaciones sobre los datos obtenidos des del servicio de Peticiones a la API.

Un ejemplo sería devolver la lista de cromos del alumno sin repeticiones, es decir, una solo copia del cromo pero con una constante que nos indique cuantas repeticiones tiene el alumno de ese cromo. Lógicamente, primero se hace una consulta a la API para obtener todos los cromos del alumno, y luego se llama a una función desarrollada en el servicio de cálculos llamada *GenerarListaSinRepetidos* (**Figura 6.4**) para que nos devuelva otra lista sin cromos repetidos y esta es la que llamamos des del *typescript* del componente de la pantalla donde se muestran los cromos del alumno.

```
GeneraListaSinRepetidos(listaCromos: Cromo []): any [] {
  const listaCromosSinRepetidos: any [] = [];
  // tslint:disable-next-line:prefer-for-of
  for (let i = 0; i < listaCromos.length; i++) {
    const n = listaCromos.filter (cromo => cromo.Nombre === listaCromos[i].Nombre).length;
    if (listaCromosSinRepetidos.filter (res => res.cromo.Nombre === listaCromos[i].Nombre).length === 0) {
      listaCromosSinRepetidos.push ({rep: n, cromo: listaCromos[i]});
    }
  }
  return listaCromosSinRepetidos;
}
```

Figura 6.4 Función del servicio de Cálculos

6.3. Sesión

El servicio de sesión ha sido muy útil durante el desarrollo de esta aplicación porque ha servido para reutilizar constantes obtenidas de la base de datos en un momento y ya no ha hecho falta volverlas a buscar porque ya estaban guardadas en la Sesión. Un ejemplo es cuando el alumno inicia sesión en la aplicación y entonces se guarda la constante Alumno llamando a la función *TomaAlumno* (**Figura 6.5**). Una vez guardada nos puede ser útil en la siguiente pantalla, por ejemplo, para buscar los juegos que tiene el alumno. Entonces llamamos a la función *DameAlumno* (**Figura 6.5**) de la Sesión para que posteriormente podamos hacer una consulta a la API y buscar los juegos de este alumno en concreto.

```
public TomaAlumno(alumno: Alumno) {
  this.alumno = alumno;
}
public DameAlumno(): Alumno {
  return this.alumno;
}
```

Figura 6.5 Funciones del servicio de Sesión

6.4. Componentes de la aplicación

Los componentes están compuestos por tres tipos de archivos:

- El archivo *Typescript* que es el que contiene los datos que posteriormente visualizaremos en la pantalla y la lógica del componente.
- El archivo *Html* que es el que contiene la arquitectura y la interfaz gráfica.
- El archivo *Scss* que es donde se definen los estilos gráficos como por ejemplo modificar la estética de una tabla o un botón.

Esta aplicación dispone de 10 componentes de tipo pantalla como por ejemplo el componente información que es la pantalla donde se muestra la información de los juegos.

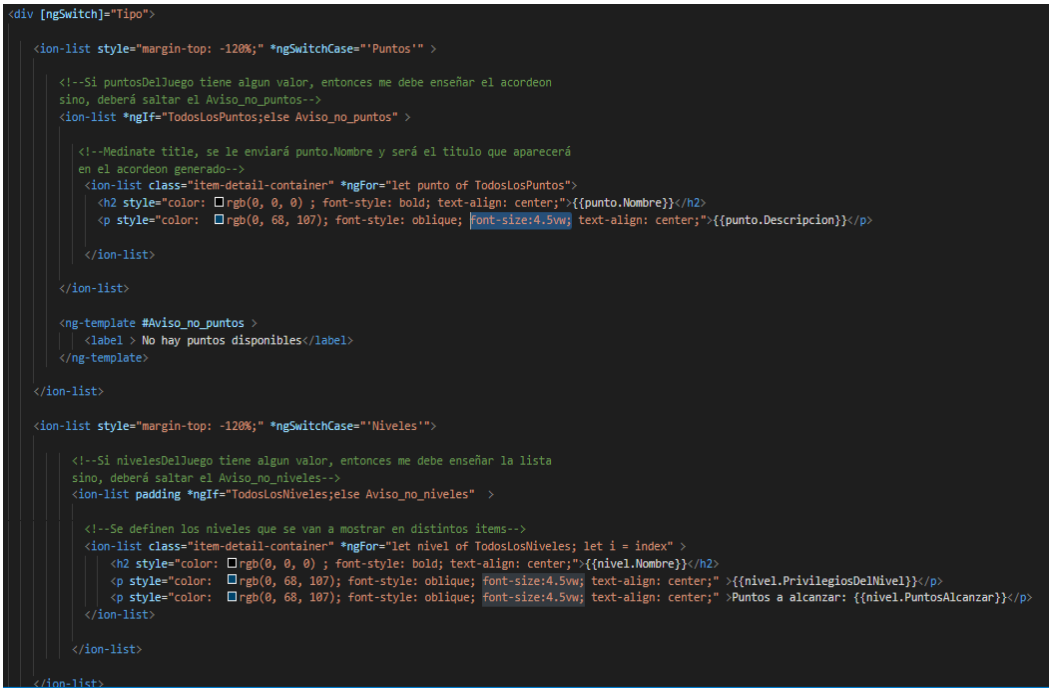
En la **Figura 6.6** se muestra el archico Typescript de la pantalla de información. Después en la **Figuras 6.7** y **6.8** se muestran los archivos de Html y Scss correspondientes.

```
export class InformacionPage implements OnInit {

  juegoSeleccionado: Juego;
  TodosLosNiveles: Nivel[] = [];
  TodosLosPuntos: Punto[] = [];
  constructor(
    private sesion: SesionService,
    private peticionesAPI: PeticionesAPIService,
    private calculos: CalculosService,
  ) { }

  ngOnInit() {
    this.juegoSeleccionado = this.sesion.DameJuego();
    this.peticionesAPI.DameNivelesJuegoDePuntos(this.juegoSeleccionado.id).subscribe(
      niveles => {
        this.TodosLosNiveles = niveles;
        console.log(this.TodosLosNiveles);
        this.peticionesAPI.DamePuntosJuegoDePuntos(this.juegoSeleccionado.id).subscribe(
          puntos => {
            this.TodosLosPuntos = puntos;
            console.log(this.TodosLosPuntos);
          }
        );
      }
    );
  }
```

Figura 6.6 Archivo *Typescript*



```
<div [ngSwitch]="Tipo">

    <ion-list style="margin-top: -120%;" *ngSwitchCase="'Puntos'" >

        <!--Si puntosDelJuego tiene algun valor, entonces me debe enseñar el acordeon
        sino, deberá saltar el Aviso_no_puntos-->
        <ion-list *ngIf="TodosLosPuntos;else Aviso_no_puntos" >

            <!--Medinate title, se le enviará punto.Nombre y será el titulo que aparecerá
            en el acordeon generado-->
            <ion-list class="item-detail-container" *ngFor="let punto of TodosLosPuntos">
                <h2 style="color: rgb(0, 0, 0) ; font-style: bold; text-align: center;">{{punto.Nombre}}</h2>
                <p style="color:  rgb(0, 68, 107); font-style: oblique; font-size:4.5vw; text-align: center;">{{punto.Descripcion}}</p>

            </ion-list>

        </ion-list>

        <ng-template #Aviso_no_puntos >
            <label > No hay puntos disponibles</label>
        </ng-template>

    </ion-list>

    <ion-list style="margin-top: -120%;" *ngSwitchCase="'Niveles'">

        <!--Si nivelesDelJuego tiene algun valor, entonces me debe enseñar la lista
        sino, deberá saltar el Aviso_no_niveles-->
        <ion-list padding *ngIf="TodosLosNiveles;else Aviso_no_niveles"  >

            <!--Se definen los niveles que se van a mostrar en distintos items-->
            <ion-list class="item-detail-container" *ngFor="let nivel of TodosLosNiveles; let i = index" >
                <h2 style="color: rgb(0, 0, 0) ; font-style: bold; text-align: center;">{{nivel.Nombre}}</h2>
                <p style="color:  rgb(0, 68, 107); font-style: oblique; font-size:4.5vw; text-align: center;" >{{nivel.PrivilegiosDelNivel}}</p>
                <p style="color:  rgb(0, 68, 107); font-style: oblique; font-size:4.5vw; text-align: center;" >Puntos a alcanzar: {{nivel.PuntosAlcanzar}}</p>
            </ion-list>

            </ion-list>

    </ion-list>
```

Figura 6.7 Archivo *Html*

```
.item-detail-container {
    z-index: -1;
    border-radius: 10px;
    position: relative;
    // margin: 1rem;
    font-family: Righteous;
    width: 100%;
    height: 130px;
    max-width: 100%;
    max-height: 130px;
    background: rgb(191, 234, 240);
    border-radius: 10px;
    border-top-width: 30px;
    border-color: darken($color: #000000, $amount: 4);
    overflow: hidden;
    // margin-left: 5%;
    margin-top: 1%;
    text-align: start;
    box-shadow: 2px 1px 2px 2px rgba(0, 0, 0, 0.25);
}
ion-toolbar {
    --ion-background-color: rgb(0, 68, 107);
  }
  .ClasspipCabeceraimg{
    position:relative;
    width:30px;
    height:30px;
    margin-left: 10%;
    margin-right: -25%;
    margin-top: -10%;
    margin-bottom: 3%;
}
```

Figura 6.8 Archivo *Scs*

CAPÍTULO 7. PRUEBAS Y EVALUACIÓN

Una vez finalizado todo el desarrollo, hay que comprobar que la aplicación sea funcional y robusta. Hay que asegurarse que no salten errores de código en caso de que el usuario se equivoque en algún punto. Para ello, se han hecho una serie de pruebas que se citan a continuación.

7.1. Lista de pruebas

- **Usuario y/o contraseña incorrectos**: El usuario se puede equivocar introduciendo sus credenciales al iniciar la sesión, así que lo más conveniente es que cuando así sea la aplicación no se bloquee y pueda mostrar el mensaje indicado tal y como se muestra en la **Figura 7.1**.

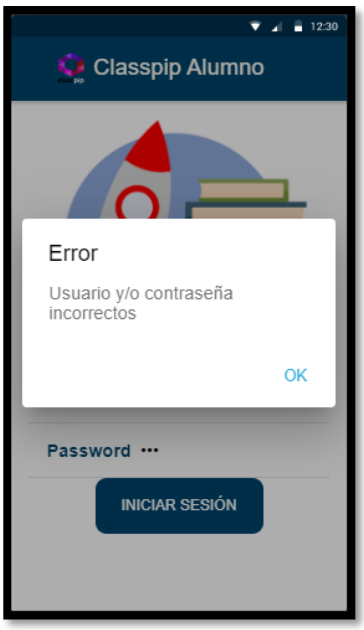


Figura 7.1 Mensaje de error al iniciar sesión

- **Aviso de falta de juegos**: Si el alumno accede a sus juegos activos o inactivos y le aparece una pantalla blanca sin ningún tipo de mensaje, puede crear confusión y creer que la aplicación funciona mal cuando en realidad el alumno no tiene ningún juego activo o inactivo. Así que para comprobar que salgan los mensajes adecuados cuando esto pase, hemos hecho la prueba de eliminar todos los juegos activos de un alumno des del Dashboard y acceder a la aplicación y efectivamente muestra el mensaje correcto tal y como se ve en la **Figura 7.2**.

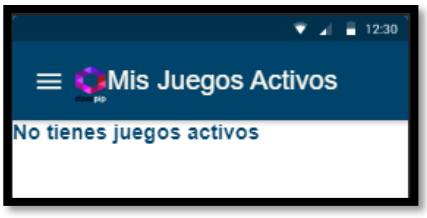


Figura 7.2 Mensaje de falta de juegos

- **Asignación de cromo**: A la hora de intercambiar el cromo con otro alumno, el usuario se puede equivocar y darle al botón de asignar el cromo sin haber elegido bien el alumno o el cromo. Así que se ha programado el código para que se muestre en mensaje idóneo para cada situación tal y como se muestra en la **Figura 7.3**.

Figura 7.3 Mensaje de error al asignar un cromo

En el caso de que se asigne el cromo correctamente también debe aparecer un mensaje de información (**Figura 7.4**). Además, se comprueba manualmente, en el álbum del alumno, que ese cromo ya no está y que lo tiene el alumno a quién lo ha cedido.

Figura 7.4 Mensaje de asignación correcta de cromo

7.2. Evaluación

Para poder sacar unas buenas conclusiones y sobre todo que sean objetivas, es bueno conocer la opinión de personas externas. Para ello, se ha mostrado la aplicación a dos perfiles de personas diferentes. El primero es una estudiante de secundaria que es el perfil a quién va dirigida la aplicación. En cambio, el segundo perfil es un conocedor del proyecto de Classpip porque es la compañera de titulación que está trabajando en una ampliación del Classpip, también, para su trabajo de fin de grado.

Una vez se haya dejado que naveguen por la aplicación libremente, las opiniones son las siguientes:

- Según la alumna de secundaria, la aplicación móvil dirigida al alumno es una herramienta muy importante porque despierta el interés del alumno por sus avances en las asignaturas y además facilita la interacción entre el alumno y el profesor mejorando así la relación entre ambos. También destaca que la aplicación es muy intuitiva y fácil de usar y destaca que el menú lateral es todo un acierto. Por otro lado, cree que usar el apellido como contraseña hace que la aplicación sea poco segura y que se habilite una función para poder cambiarla. También comenta que sería útil que en los juegos por equipo se muestre de otro color el equipo al que pertenece el alumno. Finalmente, explica que le gustaría poder usar esta aplicación en un entorno real en su clase junto a sus compañeros porque piensa que realmente sería un éxito.

- Según la compañera de titulación, piensa que la aplicación se ha integrado perfectamente en el proyecto del Classpip porque en caso de haber alguna ampliación en el Dashboard sería sencillo aplicar los cambios a la aplicación del alumno. De entre los aspectos a mejorar, destaca que sería interesante que se vean cuáles son los alumnos que forman cada equipo y además que en cada juego en el que participe el alumno se vea a qué asignatura pertenece.

Según estas opiniones, la primera versión de la aplicación del alumno para el Classpip es una versión bastante funcional, intuitiva y con mucha utilidad para el alumno. También cabe destacar que hace falta añadir pequeñas mejoras para completarla del todo y mostrar toda la información posible para el alumno.

CAPÍTULO 8. CONCLUSIONES

A continuación, se redactan una serie de conclusiones una vez finalizado el proyecto. Diferenciaré entre dos tipos de conclusiones, primero presentaré unas conclusiones técnicas que básicamente es una comparación entre el proyecto real obtenido y los objetivos que presenté en el capítulo 3. Después, añadiré unas conclusiones más a nivel personal de cómo ha sido desarrollar un proyecto de esta magnitud y usar un entorno de desarrollo totalmente desconocido hasta que he empezado a desarrollar el proyecto. Y finalmente, presentaré algunas propuestas de mejora que se pueden aplicar en el futuro.

8.1. Conclusiones técnicas

El objetivo principal del proyecto era desarrollar desde cero una aplicación móvil para el rol del alumno para completar la herramienta del Classpip. Para llegar a cumplir con este objetivo, era muy importante familiarizarse con la tecnología de desarrollo del código que hasta entonces era desconocida para mí. Después de conseguir entender el código, me he puesto varios objetivos específicos que cumplir en esta primera versión de la aplicación.

Así que llegados a este punto, puedo decir que he cumplido con todos excepto con uno:

- Poder cambiar pequeños detalles como la contraseña o la imagen de perfil del alumno.

La razón por la que no he cumplido con el objetivo de cambiar la contraseña es que como al principio se usa el apellido de contraseña, cambiarla no era tan sencillo como substituir este valor ya que el apellido se usa para mostrar los rankings de todos los juegos. Entonces poder cambiar la contraseña, implicaba modificar la clase Alumno en la base de datos del Classpip y eso tendría también implicaciones en los otros módulos como el Dashboard y la aplicación del profesor. En cambio, cambiar la foto de perfil sí que sería más sencillo, pero como se dejó para el final, tuve problemas con el programa de Cordova para hacer simulaciones en el ordenador.

8.2. Posibles mejoras en el futuro

Esta primera versión de la aplicación del alumno deja muchas posibles mejoras que llevar a cabo. A continuación, se citan algunas de ellas:

- Poder cambiar la contraseña y la foto de perfil del alumno.

- Destacar el equipo al que pertenece el alumno en un juego en concreto para que se vea claramente que es su equipo.

- Poder visualizar a qué grupo pertenece cada juego activo que tenga el alumno.

- Poder ver el detalle de los puntos obtenidos también en los juegos en equipo.

- Habilitar la función de intercambiar cromos para los juegos de colección en equipo también.

8.3. Conclusiones personales

Personalmente, llevar a cabo este proyecto ha sido todo un desafío ya que des del principio sabía que se iba a usar un lenguaje de programación que nunca se había usado durante el grado. Si es cierto que la temática de gamificación no está muy relacionada al grado de Sistemas Aeroespaciales, pero gran parte del proyecto realmente ha sido desarrollo de código y creo que eso es muy importante para un ingeniero. También me decanté por este trabajo porque viendo el mercado laboral, muchas empresas valoran más a los ingenieros que saben más lenguajes de programación. Además, las aplicaciones móviles cada vez forman más parte de nuestras vidas y actualmente hay aplicaciones para todo tipo de servicios y saber como se desarrollan estas aplicaciones me ha dado una visión diferente a la de solo como consumidora final de las aplicaciones, sino que ahora pienso en todo lo que hay detrás y todo lo que supone para un ingeniero desarrollar una aplicación desde cero.

Otra parte importante que me ha hecho crecer como ingeniera también es la de saber planificar bien los tiempos y organizar todo un trabajo que me llevaría meses acabarlo. Durante este tiempo, puedo decir que he aprovechado bien las herramientas que me han sido facilitadas por el profesor para aprender a programar con este lenguaje de manera autónoma y también he sabido enfrentarme a problemas durante el desarrollo de la aplicación, sabiendo buscar la solución correcta en internet haciendo búsquedas selectivas. Es decir, he sabido aprovechar bien la información y aplicarla a mi situación personal tal y como debe hacer un ingeniero.

Uno de los retos más grandes durante el desarrollo ha sido usar la nueva versión 4 de Ionic ya que no se había usado en el proyecto anterior y a la hora de aprovechar algunas partes del código he tenido muchos problemas con las librerías y también con los archivos *html*. Pero con la ayuda del tutor y buscando un poco más por internet, he podido solventar la mayoría de esos problemas.

En la parte final del proyecto, cuando ya tenía la aplicación totalmente funcional y solo faltaba estilizarla pensé que sería un trabajo rápido y fácil pero fue todo lo contrario. Al decorar la aplicación por código, realmente ha sido una tarea larga y lenta porque con cada cambio que hacía, tenia que volver a compilar todo el código para ver como queda en la pantalla.

Para acabar, puedo decir que estoy satisfecha con el trabajo realizado ya que he conseguido cumplir la mayoría de los objetos que me he propuesto al principio y además estoy segura que ha sido una experiencia que me ha enriquecido mucho como persona y como ingeniera y seguro que me servirá en el futuro y en mi vida laboral.

CAPÍTULO 9. ANEXOS

Para poder iniciar la aplicación desarrollada en este proyecto hace falta instalarse toda una serie de programas. A continuación, se explican todos los pasos a seguir para llegar a ejecutar esta aplicación.

9.1. Instalación de las tecnologías

El primer paso es instalarse todos los programas correspondientes. Para ello, hay que visitar el siguiente link donde se explican todos los pasos para cualquier sistema operativo, ya sea Windows, Linux o macOs: http://classpiponboarding.herokuapp.com/developer/install

9.2. Instalación de los servicios de la API

Después de tener todos los programas instalados, ya se puede instalar todo el contenido de la base de datos para que luego la aplicación tenga los datos correctos para mostrar. La base de datos se puede descargar des de un repositorio de GitHub y estos son los pasos a seguir:

1. Abrir un terminal de comandos

2. Ejecutar el comando **npm install -g strongloop** para instalar el strogloop.

3. Ubicarse en la carpeta donde queremos que se guarde el proyecto a través del comando **cd**. Por ejemplo: **cd C:\Users\MiUsuario\Desktop\ProyectoMaster**

4. Clonar el repositorio apuntando al repositorio donde este el proyecto, el comando es el siguiente: **git clone https://github.com/sajda16/classpip-services**

5. Una vez se haya completado el comando anterior, se creará una carpeta en la carpeta ProyectoMaster llamada Classpip-Services como el nombre del repositorio. Entonces accedemos a la carpeta mediante el comando **cd Classpip-Services**.

6. Ejecutar el comando **npm install**.

7. Ejecutar el comando **node.** Para que se muestre la URL para acceder a la API.

Comentar que la instalación se hace solo la primera vez, así que en el futuro cuando se quiera acceder a la API, solo hace falta abrir un terminal situándose en la carpeta del proyecto y ejecutar el comando **npm run start**.

9.3. Instalación de la aplicación del alumno

Ahora que ya tenemos la base de datos funcionando correctamente, podemos seguir con los pasos para instalarse la aplicación móvil.

1. Repetir los pasos 1 y 3 del apartado 9.2.

2. Ejecutar el comando: **git clone https://github.com/sajda16/AppAlumno**

3. Accedemos a la carpeta que se ha creado mediante el comando **cd AppAlumno** y ejecutamos el comando **npm install**.

4. Finalmente, compilamos el proyecto mediante el comando I**onic Serve –lab**. Al cabo de unos segundos, se abrirá una ventana de navegación con la URL localhost:8200.